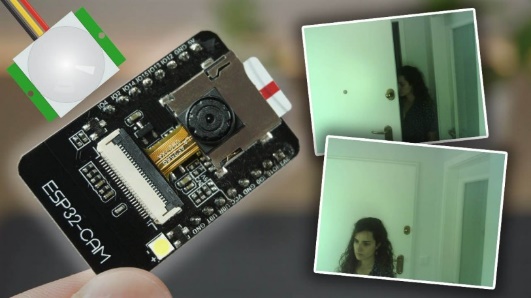
**ESP32-CAM PIR Motion Detector with Photo Capture (saves to microSD card)**

In this project, we’re going to make a motion sensor detector with photo capture using an ESP32-CAM. When your PIR sensor detects motion, it wakes up, takes a photo and saves it in the microSD card.



This project [is very similar with a previous one](https://randomnerdtutorials.com/esp32-cam-take-photo-save-microsd-card/), but after so many requests, we added a PIR motion sensor to the circuit. So, when motion is detected a picture is taken and saved on the microSD card.

Other ESP32-CAM projects and tutorials:

* [ESP32-CAM Video Streaming and Face Recognition with Arduino IDE](https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/)
* [ESP32-CAM Video Streaming Web Server (Home Assistant, Node-RED, etc…)](https://randomnerdtutorials.com/esp32-cam-video-streaming-web-server-camera-home-assistant/)
* [ESP32-CAM Take Photo and Save to MicroSD Card](https://randomnerdtutorials.com/esp32-cam-take-photo-save-microsd-card/)
* [ESP32-CAM Troubleshooting Guide](https://randomnerdtutorials.com/esp32-cam-troubleshooting-guide/)

We have a similar project using a Raspberry Pi and a camera module:

* [Raspberry Pi Motion Detector with Photo Capture](https://randomnerdtutorials.com/raspberry-pi-motion-detector-photo-capture/)

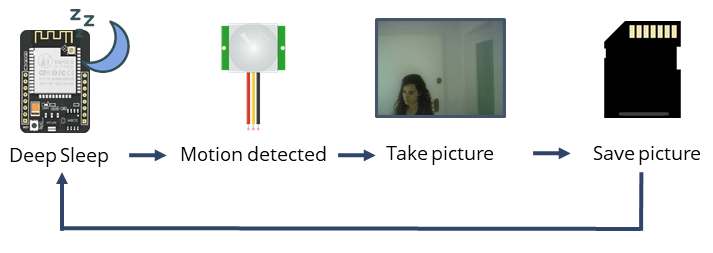
**Parts Required**

For this project, you’ll need the following parts:

* [**ESP32-CAM with OV2640**](https://makeradvisor.com/tools/esp32-cam/)
* [MicroSD card](https://makeradvisor.com/tools/microsd-card-raspberry-pi-16gb-class-10/)
* [PIR motion sensor](https://makeradvisor.com/tools/mini-hc-sr505-pir-motion-sensor/)
* [2N3904 transistor](https://makeradvisor.com/tools/transistors-npn-pnp-kit/)
* [FTDI programmer](https://makeradvisor.com/tools/ftdi-programmer-board/)
* [Female-to-female jumper wires](https://makeradvisor.com/tools/jumper-wires-kit-120-pieces/)
* [5V power supply for ESP32-CAM](https://makeradvisor.com/tools/raspberry-pi-power-supply/) or power bank (optional)

You can use the preceding links or go directly to [MakerAdvisor.com/tools](https://makeradvisor.com/tools/?utm_source=rnt&utm_medium=post&utm_campaign=post) to find all the parts for your projects at the best price!

**Project Overview**



Here is a quick overview on how the project works.

* The ESP32-CAM is in [deep sleep mode with external wake up](https://randomnerdtutorials.com/esp32-external-wake-up-deep-sleep/) enabled.
* When motion is detected, the [PIR motion sensor](https://randomnerdtutorials.com/esp32-pir-motion-sensor-interrupts-timers/) sends a signal to wake up the ESP32.
* The ESP32-CAM takes a photo and saves it on the microSD card.
* It goes back to deep sleep mode until a new signal form the PIR motion sensor is received.

**IT IS NOT NEEDED THE LOOP SECTION, BECAUSE WHEN ESP32 DETECTS A MOVE**

* **IT WILL WAKE UP**
* **IT WILL TAKE A PICTURE AND STORE**
* **IT WILL GO TO SLEEP AGAIN**

**Recommended reading:** [ESP32 Deep Sleep with Arduino IDE and Wake Up Sources](https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/)

**Formatting MicroSD Card**

The first thing we recommend doing is formatting your microSD card. You can use the Windows formatter tool or any other microSD formatter software.

|  |  |
| --- | --- |
| **1.** Insert the microSD card in your computer. Go to **My Computer** and right click in the SD card. Select **Format** |  |
| **2.**A new window pops up. Select **FAT32**, press **Start** to initialize the formatting process and follow the onscreen instructions. |  |

**Note:**according to the product specifications, the ESP32-CAM should only support 4 GB SD cards. However, we’ve tested with 16 GB SD card and it works well.

/\*\*\*\*\*\*\*\*\*

Rui Santos

Complete project details at https://RandomNerdTutorials.com/esp32-cam-pir-motion-detector-photo-capture/

IMPORTANT!!!

- Select Board "AI Thinker ESP32-CAM"

- GPIO 0 must be connected to GND to upload a sketch

- After connecting GPIO 0 to GND, press the ESP32-CAM on-board RESET button to put your board in flashing mode

Permission is hereby granted, free of charge, to any person obtaining a copy

of this software and associated documentation files.

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

\*\*\*\*\*\*\*\*\*/

#include "esp\_camera.h"

#include "esp\_timer.h"

#include "img\_converters.h"

#include "Arduino.h"

#include "fb\_gfx.h"

#include "fd\_forward.h"

#include "fr\_forward.h"

#include "FS.h" // SD Card ESP32

#include "SD\_MMC.h" // SD Card ESP32

#include "soc/soc.h" // Disable brownour problems

#include "soc/rtc\_cntl\_reg.h" // Disable brownour problems

#include "dl\_lib.h"

#include "driver/rtc\_io.h"

#include <EEPROM.h> // read and write from flash memory

// define the number of bytes you want to access

#define EEPROM\_SIZE 1

RTC\_DATA\_ATTR int bootCount = 0;

// Pin definition for CAMERA\_MODEL\_AI\_THINKER

#define PWDN\_GPIO\_NUM 32

#define RESET\_GPIO\_NUM -1

#define XCLK\_GPIO\_NUM 0

#define SIOD\_GPIO\_NUM 26

#define SIOC\_GPIO\_NUM 27

#define Y9\_GPIO\_NUM 35

#define Y8\_GPIO\_NUM 34

#define Y7\_GPIO\_NUM 39

#define Y6\_GPIO\_NUM 36

#define Y5\_GPIO\_NUM 21

#define Y4\_GPIO\_NUM 19

#define Y3\_GPIO\_NUM 18

#define Y2\_GPIO\_NUM 5

#define VSYNC\_GPIO\_NUM 25

#define HREF\_GPIO\_NUM 23

#define PCLK\_GPIO\_NUM 22

int pictureNumber = 0;

#define uS\_TO\_S\_FACTOR 1000000

void setup() {

WRITE\_PERI\_REG(RTC\_CNTL\_BROWN\_OUT\_REG, 0); //disable brownout detector

Serial.begin(115200);

Serial.setDebugOutput(true);

camera\_config\_t config;

config.ledc\_channel = LEDC\_CHANNEL\_0;

config.ledc\_timer = LEDC\_TIMER\_0;

config.pin\_d0 = Y2\_GPIO\_NUM;

config.pin\_d1 = Y3\_GPIO\_NUM;

config.pin\_d2 = Y4\_GPIO\_NUM;

config.pin\_d3 = Y5\_GPIO\_NUM;

config.pin\_d4 = Y6\_GPIO\_NUM;

config.pin\_d5 = Y7\_GPIO\_NUM;

config.pin\_d6 = Y8\_GPIO\_NUM;

config.pin\_d7 = Y9\_GPIO\_NUM;

config.pin\_xclk = XCLK\_GPIO\_NUM;

config.pin\_pclk = PCLK\_GPIO\_NUM;

config.pin\_vsync = VSYNC\_GPIO\_NUM;

config.pin\_href = HREF\_GPIO\_NUM;

config.pin\_sscb\_sda = SIOD\_GPIO\_NUM;

config.pin\_sscb\_scl = SIOC\_GPIO\_NUM;

config.pin\_pwdn = PWDN\_GPIO\_NUM;

config.pin\_reset = RESET\_GPIO\_NUM;

config.xclk\_freq\_hz = 20000000;

config.pixel\_format = PIXFORMAT\_JPEG;

pinMode(4, INPUT);

digitalWrite(4, LOW);

rtc\_gpio\_hold\_dis(GPIO\_NUM\_4);

if(psramFound()){

config.frame\_size = FRAMESIZE\_UXGA; // FRAMESIZE\_ + QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA

config.jpeg\_quality = 10;

config.fb\_count = 2;

} else {

config.frame\_size = FRAMESIZE\_SVGA;

config.jpeg\_quality = 12;

config.fb\_count = 1;

}

// Init Camera

esp\_err\_t err = esp\_camera\_init(&config);

if (err != ESP\_OK) {

Serial.printf("Camera init failed with error 0x%x", err);

return;

}

Serial.println("Starting SD Card");

delay(500);

if(!SD\_MMC.begin()){

Serial.println("SD Card Mount Failed");

//return;

}

uint8\_t cardType = SD\_MMC.cardType();

if(cardType == CARD\_NONE){

Serial.println("No SD Card attached");

return;

}

// Take Picture with Camera

camera\_fb\_t \* fb = NULL;

fb = esp\_camera\_fb\_get();

if(!fb) {

Serial.println("Camera capture failed");

return;

}

// initialize EEPROM with predefined size

EEPROM.begin(EEPROM\_SIZE);

pictureNumber = EEPROM.read(0) + 1;

// Path where new picture will be saved in SD Card

String path = "/picture" + String(pictureNumber) +".jpg";

fs::FS &fs = SD\_MMC;

Serial.printf("Picture file name: %s\n", path.c\_str());

File file = fs.open(path.c\_str(), FILE\_WRITE);

if(!file){

Serial.println("Failed to open file in writing mode");

}

else {

file.write(fb->buf, fb->len); // payload (image), payload length

Serial.printf("Saved file to path: %s\n", path.c\_str());

EEPROM.write(0, pictureNumber);

EEPROM.commit();

}

file.close();

esp\_camera\_fb\_return(fb);

delay(1000);

// Turns off the ESP32-CAM white on-board LED (flash) connected to GPIO 4

pinMode(4, OUTPUT);

digitalWrite(4, LOW);

rtc\_gpio\_hold\_en(GPIO\_NUM\_4);

esp\_sleep\_enable\_ext0\_wakeup(GPIO\_NUM\_13, 0);

Serial.println("Going to sleep now");

delay(1000);

esp\_deep\_sleep\_start();

Serial.println("This will never be printed");

}

void loop() {

}

This code is very similar to one of our previous ESP32-CAM projects, but it enables external wake up on GPIO 13.

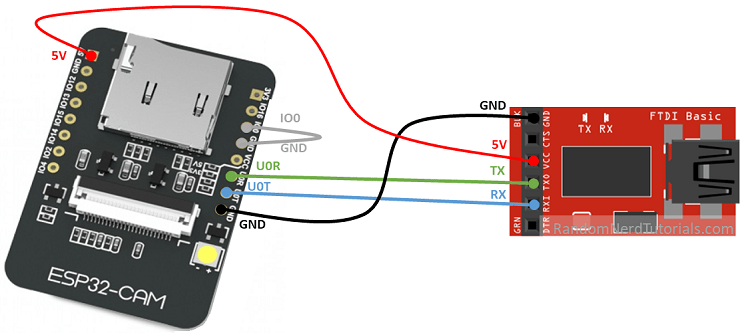
esp\_sleep\_enable\_ext0\_wakeup(GPIO\_NUM\_13, 0);

To learn more about the code, go to the following project:

* [ESP32-CAM Take Photo and Save to MicroSD Card](https://randomnerdtutorials.com/esp32-cam-take-photo-save-microsd-card/)

**ESP32-CAM Upload Code**

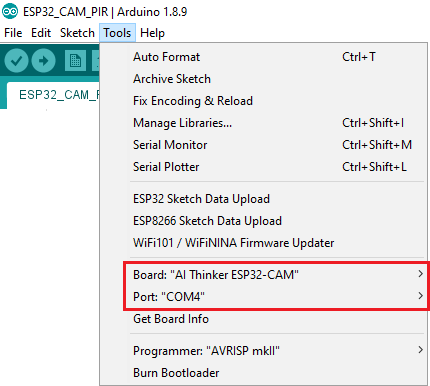
To upload code to the ESP32-CAM board, connect it to your computer using an [FTDI programmer](https://makeradvisor.com/tools/ftdi-programmer-board/). Follow the next schematic diagram:

[](https://i2.wp.com/randomnerdtutorials.com/wp-content/uploads/2019/08/ESP32-CAM-FTDI-programmer-5V-supply.png?ssl=1)

**Important:**GPIO 0 needs to be connected to GND so that you’re able to upload code.

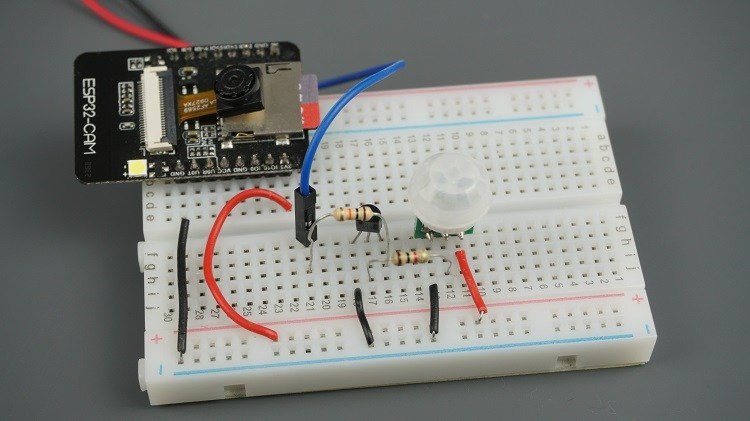
To upload the code, follow the next steps:

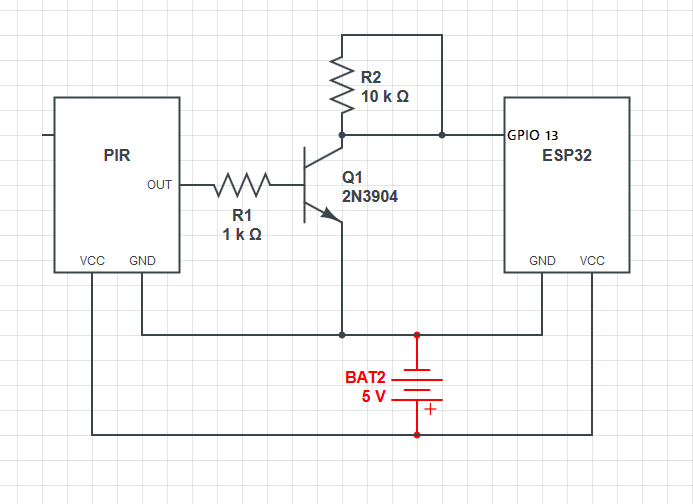
1. Go to **Tools**> **Board**and select **AI Thinker ESP32-CAM**
2. Go to **Tools**> **Port**and select the COM port the ESP32 is connected to
3. Press the ESP32-CAM on-board RESET button
4. Then, click the upload button to upload the code

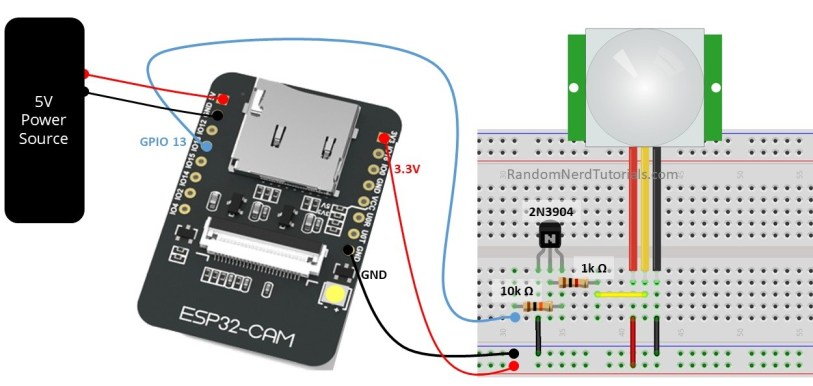


**Important:**if you can’t upload the code, double-check that GPIO 0 is connected to GND and that the RX and TX connections are correct. Check if you’ve selected the right settings in the **Tools**menu. You should also press the on-board Reset button to restart your ESP32 in flashing mode.

**ESP32-CAM PIR Sensor Schematic**



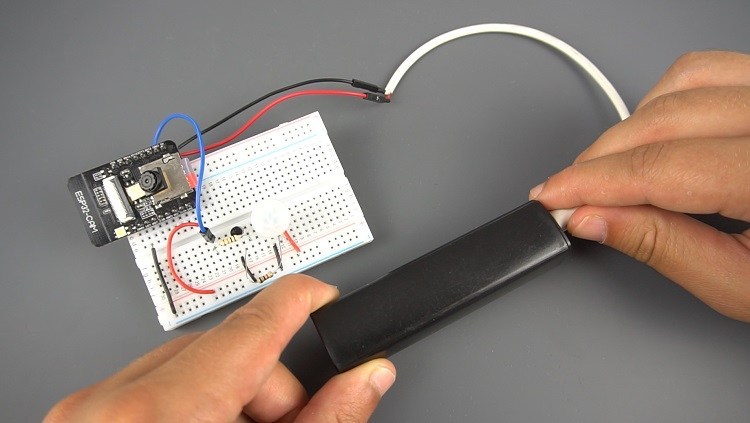
[](https://i2.wp.com/randomnerdtutorials.com/wp-content/uploads/2019/08/ESP32-CAM-PIR-motion-sensor.png?ssl=1)

[](https://i2.wp.com/randomnerdtutorials.com/wp-content/uploads/2019/08/ESP32-CAM-PIR-Motion-Sensor-Wiring.jpg?ssl=1)

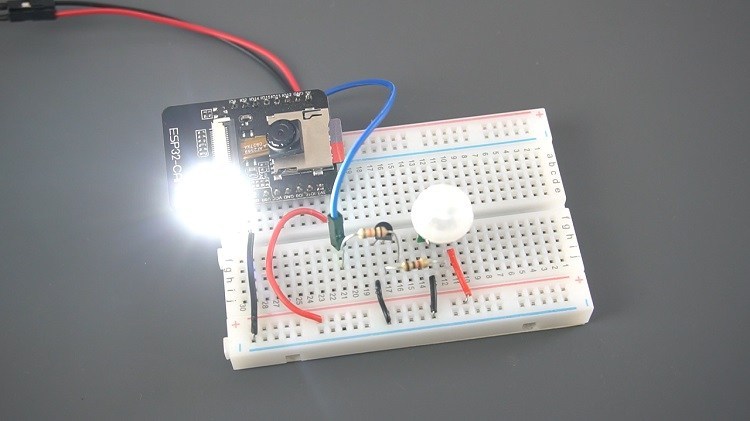
To prevent problems during upload, we recommend assembling the circuit only after uploading the code.

**Demonstration**

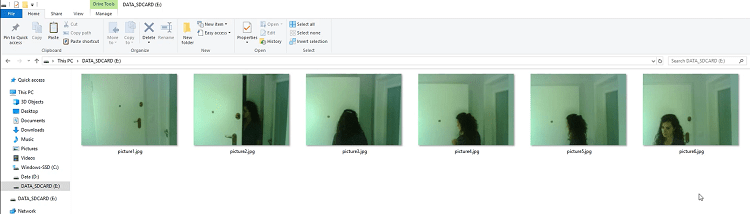
After uploading de code and assembling the circuit, insert a formatted microSD card and apply power to your circuit – you can use a portable charger, for example.



Then, press the RST button, and it should start working. When it detects motion, it turns on the flash, takes a photo and saves it on the microSD card.



Experiment with this circuit several times to make sure that it is working. Then, insert the microSD card to your computer to see the captured photos.



**Troublehsooting**

If you’re getting any of the following errors, read our [**ESP32-CAM Troubleshooting Guide: Most Common Problems Fixed**](https://randomnerdtutorials.com/esp32-cam-troubleshooting-guide/)

* Failed to connect to ESP32: Timed out waiting for packet header
* Camera init failed with error 0x20001 or similar
* Brownout detector or Guru meditation error
* Sketch too big error – Wrong partition scheme selected
* Board at COMX is not available – COM Port Not Selected
* Psram error: GPIO isr service is not installed
* Weak Wi-Fi Signal
* No IP Address in Arduino IDE Serial Monitor
* Can’t open web server
* The image lags/shows lots of latency

**Wrapping Up**

We hope you’ve liked this project. To catch our next ESP32-CAM projects, make sure you [subscribe to our newsletter](https://randomnerdtutorials.com/download). If you don’t have an ESP32-CAM yet, you can [get one for approximately $6](https://makeradvisor.com/tools/esp32-cam/).